The Cancer Genome Atlas, Glioblastoma Data

Author: Sean Davis
Contact: sdavis2@mail.nih.gov
Date: 2011-08-13

Table of Contents

1 Background

Approximately 85-90% of all primary central nervous system tumors arise in the brain[1]. The annual incidence of all brain tumors is about 6-7 per 100,000 persons per year with a mortality of about 5 per 100,000 persons per year. Glioblastoma multiforme is the most common brain tumor in humans, accounting for about 15% of all brain tumors, and generally affects adults, though children may also develop these tumors, also. The peak incidence is around 60 years of age. The disease is, in general, devastating with a mean survival of less than one year. Surgery and radiation are the primary therapeutic modalities though chemotherapy may be used for diffuse disease such as leptomeningeal seeding or positive CSF.

[1]See http://cancer.gov/ for details

From the Cancer Genome Atlas website:

> The Cancer Genome Atlas (TCGA) is a comprehensive and coordinated effort to accelerate our understanding of the molecular basis of cancer through the application of genome analysis technologies, including large-scale genome sequencing. TCGA is a joint effort of the National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI), two of the 27 Institutes and Centers of the National Institutes of Health, U.S. Department of Health and Human Services.
>
> TCGA started as a pilot project in 2006 to assess and validate the feasibility of a full-scale effort to systematically explore the entire spectrum of genomic changes involved in human cancers. With the success of the pilot project, TCGA now will expand its efforts to aggressively pursue 20 or more additional cancers to yield a comprehensive, rigorous and publicly accessible data set that will improve our ability to diagnose, treat and prevent cancer.

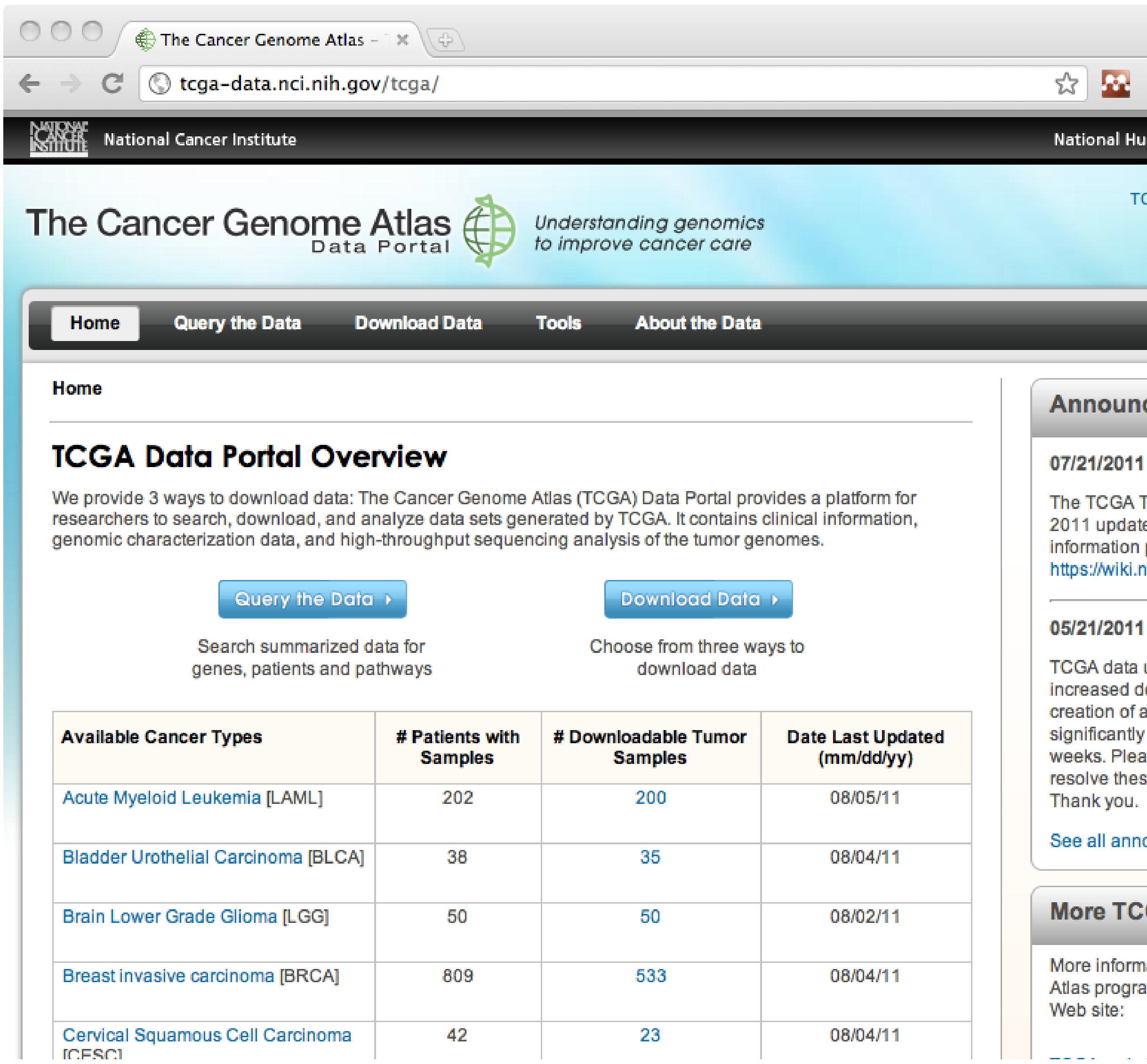
GBM is one of the tumors profiled in the pilot study. As such, there is quite a bit of profiling data available. I have compiled a subset of these data for experimentation.

Comment: Since this is the last lab of the course, *please experiment*, use the help() function liberally to learn more about the objects, classes, and functions being used. Also, if there are biological tangents to follow, please do so.

This vignette is designed to use the *TCGAGBM* data package. It should be installed prior to getting started.

2 The Data

The data for this lab are publicly available from the *TCGA data portal <http://tcga-data.nci.nih.gov/tcga/>*.


The Cancer Genome Atlas –
tcga-data.nci.nih.gov/tcga/
National Cancer Institute
National Hu
The Cancer Genome Atlas Data Portal
Understanding genomics to improve cancer care
Home
Query the Data
Download Data
Tools
About the Data
Query the Data
Download Data

Home

TCGA Data Portal Overview

We provide 3 ways to download data: The Cancer Genome Atlas (TCGA) Data Portal provides a platform for researchers to search, download, and analyze data sets generated by TCGA. It contains clinical information, genomic characterization data, and high-throughput sequencing analysis of the tumor genomes.

Search summarized data for genes, patients and pathways

Choose from three ways to download data

Available Cancer Types	# Patients with Samples	# Downloadable Tumor Samples	Date Last Updated (mm/dd/yy)
Acute Myeloid Leukemia [LAML]	202	200	08/05/11
Bladder Urothelial Carcinoma [BLCA]	38	35	08/04/11
Brain Lower Grade Glioma [LGG]	50	50	08/02/11
Breast invasive carcinoma [BRCA]	809	533	08/04/11
Cervical Squamous Cell Carcinoma [CESC]	42	23	08/04/11

Announc

07/21/2011

The TCGA T
2011 update
information
https://wiki.n

05/21/2011

TCGA data
increased d
creation of a
significantly
weeks. Plea
resolve thes
Thank you.

See all ann

More TC

More inform
Atlas progra
Web site:

Based on the availability of data for the 27k Illumina methylation platform, data were downloaded from the TCGA website. The data are available in the TCGAGBM data package in the extdata directory. The clinical data are available, also. Here is a quick overview:

```
> library(TCGAGBM)
 > clini-
cal = read.delim(system.file("extdata/Clinical/clinical_patient_public_GBM.txt.gz",
 +     package = "TCGAGBM"), header = TRUE)
 > rownames(clinical) = clinical[, 1]
 > summary(clinical)
   bcr_patient_barcode additional_chemo_therapy
 TCGA-02-0001:  1      NO  :273
 TCGA-02-0003:  1      null:157
 TCGA-02-0004:  1      YES :165
```

```
TCGA-02-0006:  1
TCGA-02-0007:  1
TCGA-02-0009:  1
(Other)     :589
additional_drug_therapy
NO  :404
null:123
YES : 68
```

```
additional_hormone_therapy
NO  :  1
null:594
```

```
additional_immuno_therapy
NO  :460
null:124
YES : 11
```

```
additional_pharmaceutical_therapy
NO  :448
null:122
YES : 25
```

```
additional_radiation_therapy
NO  :424
null:120
YES : 51
```

```
age_at_initial_pathologic_diagnosis
null   : 92
61     : 19
63     : 18
69     : 17
54     : 16
```

```
57     : 16
(Other):417
anatomic_organ_subdivision chemo_therapy
Brain:497                  NO  : 92
null : 98                  null:139
                           YES :364
```

```
days_to_birth days_to_death
null   : 92   null   :202
-21920 :  2   111    :  4
-22456 :  2   372    :  4
-23205 :  2   121    :  3
-23273 :  2   142    :  3
-24085 :  2   231    :  3
(Other):493   (Other):376
days_to_initial_pathologic_diagnosis
0   :503
null: 92
```

```
days_to_last_followup days_to_tumor_progression
null   : 92           null   :366
103    :  4           91     :  4
167    :  4           0      :  3
316    :  4           125    :  3
358    :  4           148    :  3
106    :  3           167    :  3
(Other):484           (Other):213
days_to_tumor_recurrence  drugs        gender
null   :494             null:595   FEMALE:193
427    :  3                        MALE  :310
93     :  3                        null  : 92
127    :  2
81     :  2
1013   :  1
(Other): 90
                      histological_type
null                            : 94
Treated primary GBM             : 19
Untreated primary (De Nova) GBM:  1
Untreated primary (de novo) GBM:481
```

```
hormonal_therapy immuno_therapy
NO  :412         NO  :467
null:114         null:114
YES : 69         YES : 14
```

```
informed_consent_verified
null: 65
YES :530
```

```
            initial_pathologic_diagnosis_method
EXCISIONAL BIOPSY             : 60
FINE NEEDLE ASPIRATION BIOPSY :  3
INCISION BIOPSY               :  4
null                          : 94
OTHER METHOD, SPECIFY IN NOTES:  2
TUMOR RESECTION               :432

person_neoplasm_cancer_status
null      :141
TUMOR FREE: 24
WITH TUMOR:430
```

```
pretreatment_history prior_glioma
NO  :480             NO  :487
null: 94             null: 93
YES : 21             YES : 15
```

```
radiation_therapy radiations
NO  : 60          null:595
null:111
YES :424
```

```
targeted_molecular_therapy tumor_tissue_site
NO  :407                   BRAIN:502
```

```
 null:118                  null : 93
 YES : 70




   vital_status
 DECEASED:393
 LIVING  :108
 null    : 94




 year_of_initial_pathologic_diagnosis
 null   : 92
 2009   : 67
 2008   : 66
 2005   : 53
 2006   : 39
 2007   : 37
 (Other):241
```

2.1 Methylation data preparation

The methylation data were not in any standard Illumina data dump format, so some custom code is necessary to convert to a MethyLumiSet object.

```
> tmp = sap-
ply(dir(system.file("extdata/TCGA_GBM_IlluminaMethylation27k",
 +     package = "TCGAGBM"), pattern = "Methyla-
tion27.7"),
 +     function(x) {
 +         message(x)
 +         read.delim(file.path(system.file("extdata/TCGA_GBM_IlluminaMethylation27k"
 +             package = "TCGAGBM"), x),
 +             header = TRUE, skip = 1)
 +     }, simplify = FALSE)
 > matrixnames = colnames(tmp[[1]])
 > matrixlist = lapply(matrixnames, function(x) {
 +     message(x)
 +     sapply(tmp, function(y) y[, x])
 + })
 > names(matrixlist) <- matrixnames
 > b = ma-
trixlist$Methylated_Signal_Intensity..M./(matrixlist$Methylated_Signal_Intensity..M. 
 +     matrixlist$Un.Methylated_Signal_Intensity..U. +
 +     100)
 > methTCGA = new("MethyLumiSet", betas = b)
 > for (x in names(matrixlist)) {
```

```
 +     message(x)
 +     assayDataElement(methTCGA, x) = matrixlist[[x]]
 + }
 > featureNames(methTCGA) <- assayDataElement(methTCGA,
 +     "Composite.Element.REF")[, 1]
 > sampleNames(methTCGA) = substr(sub("jhu-
usc.edu_GBM.HumanMethylation27.7.lvl-1.",
 +     "", sampleNames(methTCGA)), 1, 12)
 > pData(methTCGA) = clinical[match(clinical[,
 +     1], sampleNames(methTCGA)), ]
 > annotation(methTCGA) = "IlluminaHumanMethylation27k"
 > experimentData(methTCGA) = new("MIAME",
 +     name = "TCGA GBM Methylation 27k data",
 +     lab = "TCGA, JHU methylation", url = "http://cancergenome.nih.gov/")
```

Now, to look at what we have just loaded:

```
> methTCGA
Object Information:
MethyLumiSet (storageMode: lockedEnvironment)
assayData: 27578 features, 45 samples
  element names: betas, Composite.Element.REF, Detec-
tion_P_Value, M_Number_Beads, M_STDERR, Methy-
lated_Signal_Intensity..M., Nega-
tive_Control_Grn_Avg_Intensity, Nega-
tive_Control_Grn_STDERR, Nega-
tive_Control_Red_Avg_Intensity, Nega-
tive_Control_Red_STDERR, U_Number_Beads, U_STDERR, Un.Methylated_Signal_Intensity..U.
protocolData: none
phenoData
  sampleNames: TCGA-02-2466 TCGA-02-2470 ...
    TCGA-32-1986 (45 total)
  varLabels: BCRPATIENTBARCODE
    TUMORTISSUESITE ... X (34 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: IlluminaHumanMethylation27k
Major Operation History:
 [1] submitted finished  command
 <0 rows> (or 0-length row.names)
 > experimentData(methTCGA)
Experiment data
  Experimenter name: TCGA GBM Methylation 27k data
  Laboratory: TCGA, JHU methylation
  Contact information:
  Title:
  URL: http://cancergenome.nih.gov/
  PMIDs:
  No abstract available.
```

2.2 CGH data preparation

Data from the same patients as above were downloaded from the TCGA website. Briefly, the samples were run on 244k Agilent long oligo arrays with Promega commercial DNA run as a reference. We can use the limma package to load and manipulate the data.

```
> library(limma)
 > tmp2 = read.maimages(files = dir(system.file("extdata/TCGA_GBM_244kcgh",
 +     package = "TCGAGBM"), pattern = "MSK"),
 +     path = system.file("extdata/TCGA_GBM_244kcgh",
 +         package = "TCGAGBM"), source = "agilent")
```

A little convenience function, splitAgilentChromLocs is useful to get the chromosome locations from the feature extraction data.

```
> splitAgilentChromLocs <- function(systematicName) {
 +     tmp <- gsub("[:-
]", ":", as.character(systematicName))
 +     tmp2 <- data.frame(do.call(rbind,
 +         strsplit(as.character(tmp), ":")))
 +     colnames(tmp2) <- c("chrom", "start",
 +         "end")
 +     tmp2[, 2] = as.integer(as.character(tmp2[,
 +         2]))
 +     tmp2[, 3] = as.integer(as.character(tmp2[,
 +         3]))
 +     return(tmp2)
 + }
 > locs = splitAgilentChrom-
Locs(as.character(tmp2$genes$SystematicName))
 > tmp2$genes = data.frame(tmp2$genes, locs)
```

Remove the control probes and "normalize"--that is, convert to MA from RG.

```
> tmp2 = tmp2[tmp2$genes$ControlType ==
 +     0, ]
 > cghTCGAMA = normalizeWithinArrays(tmp2,
 +     method = "none", bc.method = "none")
```

Now, map the sample names back to the arrays.

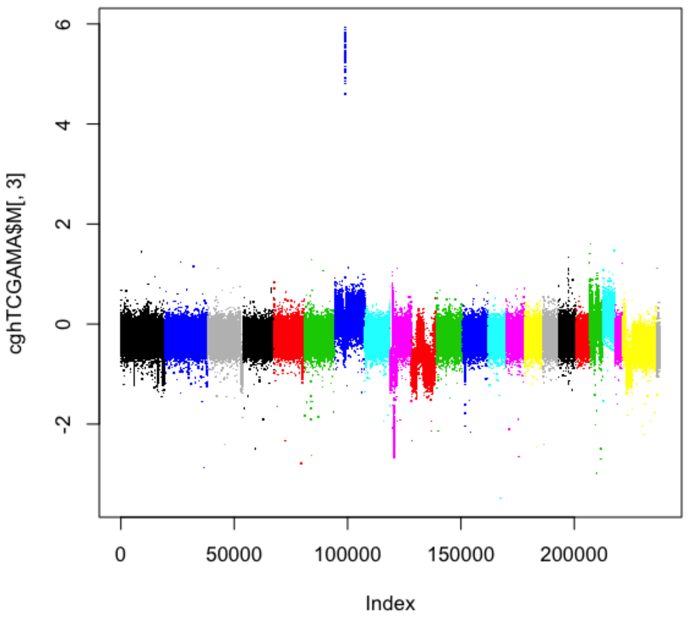
```
> cghs-
drf = read.delim(system.file("extdata/TCGA_GBM_244kcgh/mskcc.org_GBM.HG-
CGH-244A.1.sdrf.txt",
 +     package = "TCGAGBM"))
 > cghsdrf = cghsdrf[cghsdrf$Provider ==
 +     "BCR", ]
 > cghTCGAMA$targets$sample = sub-
str(cghsdrf[match(rownames(cghTCGAMA$targets),
 +     cghsdrf$Array.Data.File), 1], 1, 12)
 > cghTCGAMA$targets = data.frame(cghTCGAMA$targets,
 +     clinical[match(cghTCGAMA$targets$sample,
 +         clinical[, 1]), ])
```

And finally, just to make things easier in the future, order the probes in chromosome order and remove probes mapping to odd places like chr7_random, etc.

```
> numericchrom = sub("chr", "", cghTCGAMA$genes$chrom)
 > numericchrom[numericchrom == "X"] = 23
 > numericchrom[numericchrom == "Y"] = 24
 > numericchrom = as.integer(numericchrom)
 > cghTCGAMA = cghTCGAMA[!is.na(numericchrom),
 +     ]
 > numericchrom = numericchrom[!is.na(numericchrom)]
 > cghTCGAMA$genes$chrom = fac-
tor(as.character(cghTCGAMA$genes$chrom))
 > cghTCGAMA$genes$Chr = numericchrom
 > cghTCGAMA$genes$Position = as.numeric(cghTCGAMA$genes$start)
 > cghTCGAMA = cghTCGAMA[order(numericchrom,
 +     cghTCGAMA$genes$start), ]
```

Just to get a rough idea of how things look, let's make a plot of one of the samples:

```
> plot(cghTCGAMA$M[, 3], pch = ".", col = cghTCGAMA$genes$chrom)
```



2.3 Expression data preparation

Again, the data were downloaded from the TCGA website. A rather unusual array platform was used for the gene expression analysis for which there was actually no array description file available. The array is a custom Agilent 244k array with multiple probes per gene. However, the TCGA project does make available processed data for these samples. It appears that the processing was pretty standard with the multiple probes per gene averaged and then loess normalized. In any case, for our purposes, the data are probably good enough for comparison to other data types.

```
> dat1 = read.delim(system.file("extdata/TCGA_GBM_geneexp/unc.edu__AgilentG4502A_07_2
 +     package = "TCGAGBM"), header = TRUE)
 > datmat = matrix(as.numeric(as.character(dat1$value)),
 +     nrow = nrow(dat1)/length(unique(dat1$barcode)))
 > colnames(datmat) = substr(unique(dat1$barcode),
 +     1, 12)
 > row-
names(datmat) = dat1$gene.symbol[1:(nrow(dat1)/length(unique(dat1$barcode)))]
 > expTCGA = new("ExpressionSet", exprs = datmat)
 > experimentData(expTCGA) = new("MIAME",
 +     name = "TCGA GBM level 3 expression data",
 +     lab = "TCGA, UNC gene expression",
 +     url = "http://cancergenome.nih.gov/")
 > pData(expTCGA) = clinical[match(sampleNames(expTCGA),
 +     clinical[, 1]), ]
```

3 Analyses

Now that the data are loaded and organized, it is time to begin some analysis. Note that I supply some suggested workflows, but there are plenty of tangents that might be worth following.

3.1 Expression and Methylation Correlation

It is interesting, of course, to examine directly the effects of DNA methylation on gene expression. Armed with our DNA methylation data and our gene expression data, we can directly calculate correlations between the two data types. In order to perform the calculations, we will want to generate two matrices, one for gene expression and one for DNA methylation. Each matrix will need to have the same ordering of samples so that we can calculate on them directly. Also, we will want to map the DNA methylation probes to their matching gene expression probes. Recall that the gene expression data are in the form on an ExpressionSet and that the featureNames of the ExpressionSet are the actual HGNC gene names.

```
> data(expTCGA)
 > featureNames(expTCGA)[10:20]
 [1] "A4GNT"   "AAAS"    "AACS"    "AADAC"
 [5] "AADACL1" "AADACL2" "AADACL3" "AADACL4"
 [9] "AADAT"   "AAK1"    "AAMP"
```

For the methylation data, on the other hand, the annotation for the probes is not directly available from the MethyLumiSet object, methTCGA. Instead, the featureNames are Illumina probe names.

```
> data(methTCGA)
 > featureNames(methTCGA)[10:20]
 [1] "cg00010193" "cg00011459" "cg00012199"
 [4] "cg00012386" "cg00012792" "cg00013618"
 [7] "cg00014085" "cg00014837" "cg00015770"
 [10] "cg00016968" "cg00019495"
```

The appropriate data package is the IlluminaHumanMethylation27k.db package. It may be necessary to install it using biocLite if it is not already installed. Attempting to load the package will show if it is installed.

```
> require(IlluminaHumanMethylation27k.db)
```

As with other annotation packages, we can quickly look to see what is inside:

```
> IlluminaHumanMethylation27k()
Quality control information for IlluminaHumanMethyla-
tion27k:


This package has the following mappings:

IlluminaHumanMethylation27kACCNUM has 27551 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kALIAS2PROBE has 57835 mapped keys (of 112379 keys)
IlluminaHumanMethylation27kCHR has 25821 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kCHRLENGTHS has 93 mapped keys (of 93 keys)
IlluminaHumanMethylation27kCHRLOC has 25805 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kCHRLOCEND has 25805 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kENSEMBL has 25746 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kENSEMBL2PROBE has 14519 mapped keys (of 19948 keys)
IlluminaHumanMethylation27kENTREZID has 25821 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kENZYME has 3649 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kENZYME2PROBE has 910 mapped keys (of 975 keys)
IlluminaHumanMethylation27kGENENAME has 25821 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kGO has 24351 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kGO2ALLPROBES has 14024 mapped keys (of 14407 keys)
IlluminaHumanMethylation27kGO2PROBE has 10569 mapped keys (of 10981 keys)
IlluminaHumanMethylation27kMAP has 25763 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kOMIM has 20148 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kPATH has 9063 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kPATH2PROBE has 229 mapped keys (of 229 keys)
IlluminaHumanMethylation27kPFAM has 25799 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kPMID has 25814 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kPMID2PROBE has 284707 mapped keys (of 302146 keys)
IlluminaHumanMethylation27kPROSITE has 25799 mapped keys (of 27578 keys)
```

```
IlluminaHumanMethylation27kREFSEQ has 25821 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kSYMBOL has 25821 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kUNIGENE has 25806 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kUNIPROT has 25732 mapped keys (of 27578 keys)


Additional Information about this package:

DB schema: HUMANCHIP_DB
DB schema version: 2.1
Organism: Homo sapiens
Date for NCBI data: 2011-Mar16
Date for GO data: 20110312
Date for KEGG data: 2011-Mar15
Date for Golden Path data: 2010-Mar22
Date for IPI data: 2011-Feb18
Date for Ensembl data: 2011-Feb2
```

Since we want to map methylation and gene expression to each other, we can use the annotation package to get gene names for the methylation data.

```
> methgenenames = unlist(mget(featureNames(methTCGA),
 +     IlluminaHumanMethylation27kSYMBOL,
 +     ifnotfound = NA))
 > sum(is.na(methgenenames))
 [1] 1757
```

There are 1611 probes on the methylation platform that, for whatever reason, do not have associated gene symbols. We will simply exclude those from downstream analysis. The mapping between platforms is now pretty straightforward.

```
> tmp = match(methgenenames, featureNames(expTCGA))
 > methdat = betas(methTCGA)[!is.na(tmp),
 +     order(sampleNames(methTCGA))]
 > expdat = exprs(expTCGA)[tmp[!is.na(tmp)],
 +     order(sampleNames(expTCGA))]
```

Here, we are relying on the fact that the sample names are the same between the two data sets so that ordering by sampleNames will result in matching orders. We can quickly double-check that that is the case.

```
> match(colnames(expdat), colnames(methdat))
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
 [16] 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
 [31] 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
 > dim(expdat)
 [1] 23565    45
 > dim(methdat)
 [1] 23565    45
```

The sample names appear to match and the matrices are the same size. Now, we are ready to calculate some correlations. The statement below simply does a row wise Pearson correlation calculation. The result will be one correlation value per row of data, each of which corresponds to a methylation probe and its associated expression probe. Note that each expression probe may map to several methylation probes.

```
> x = sapply(1:nrow(methdat), function(i) cor(methdat[i,
+     ], expdat[i, ]))
```

So, what would we expect this distribution to look like?

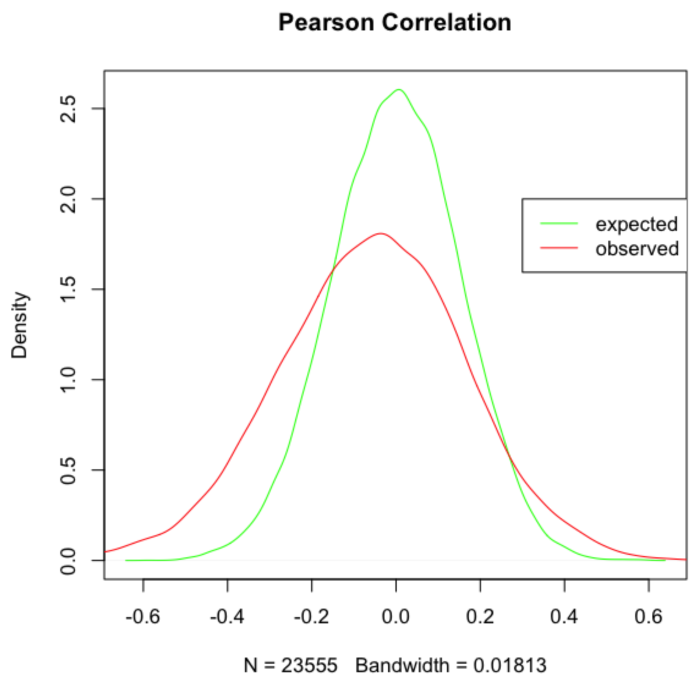
```
> summary(x)
    Min.  1st Qu.   Median     Mean  3rd Qu. 
-0.92580 -0.20600 -0.05303 -0.05888  0.09231 
    Max.     NA's 
 0.83900 10.00000 
```

The mean is less than 0 and there is a definite skew to the left. How big is the effect? In order to determine what the distribution might look like under the null hypothesis, we can simply permute one of the sample sets relative to the other, effectively breaking any correlation that might exist.

```
> y = sapply(1:nrow(methdat), function(i) cor(methdat[i,
+     ], expdat[i, sample(1:45, 45)]))
> summary(y)
     Min.   1st Qu.    Median      Mean   3rd Qu. 
-0.587100 -0.102200  0.001313  0.000238  0.102900 
     Max.      NA's 
 0.584200 10.000000 
```

Though this is only one replicate of randomization, it is obvious that this distribution looks more like what we would expect if there were no effect of DNA methylation on gene expression. A quick plot is probably useful to examine the differences more globally.

```
> plot(density(y[!is.na(y)]), col = "green",
+     main = "Pearson Correlation")
> lines(density(x[!is.na(x)]), col = "red")
> legend(0.3, 2, legend = c("expected",
+     "observed"), col = c("green", "red"),
+     lty = 1)
```



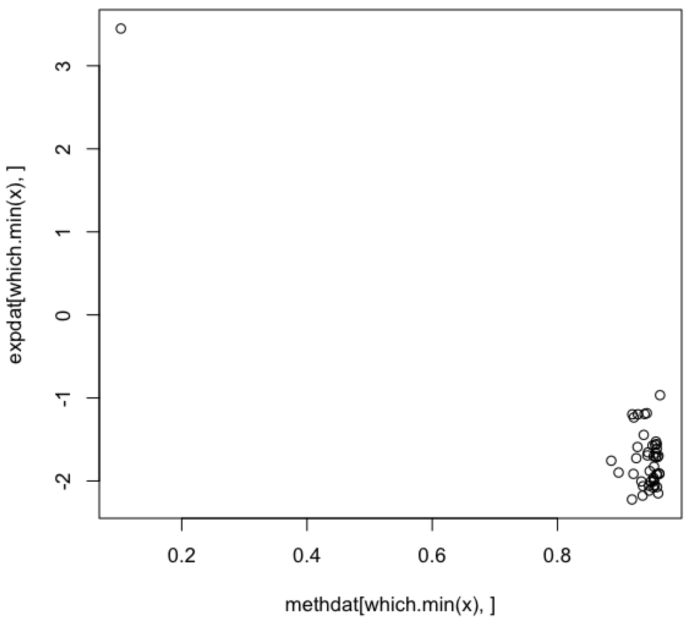
Note the shoulder to the left in the 'observed' data, indicating an effect of DNA methylation on gene expression. One could also do some hypothesis testing to determine if the effect is statistically significant.

There appears to be a global effect of DNA methylation on gene expression. Which genes show the largest effect?

```
> mincor = rownames(methdat)[which.min(x)]
 > mget(mincor, IlluminaHumanMethylation27kSYMBOL)
 $cg15933546
 [1] "DNAH8"
```

And what does a plot of gene expression and DNA methylation look like for that probe?

```
> plot(methdat[which.min(x), ], expdat[which.min(x),
 +     ])
```

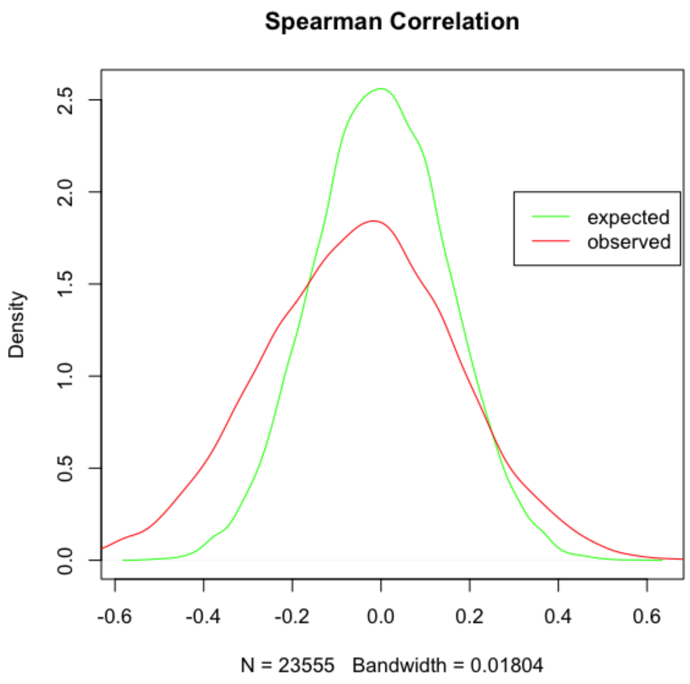


So, there appears to be a bit of a problem with our correlation measure. The Pearson correlation measure will find such outliers quite nicely. Instead, perhaps we should use a rank-based correlation metric.

```
> x = sapply(1:nrow(methdat), function(i) cor(methdat[i,
 +     ], expdat[i, ], method = "spearman"))
 > summary(x)
    Min.  1st Qu.   Median     Mean  3rd Qu.
 -0.82600 -0.19980 -0.04592 -0.05115  0.09776
    Max.     NA's
 0.72650 10.00000
> y = sapply(1:nrow(methdat), function(i) cor(methdat[i,
+     ], expdat[i, sample(1:45, 45)], method = "spear-
man"))
> summary(y)
     Min.   1st Qu.    Median      Mean   3rd Qu.
-0.528100 -0.102700 -0.000725 -0.000403  0.102700
     Max.      NA's
 0.579700 10.000000
```

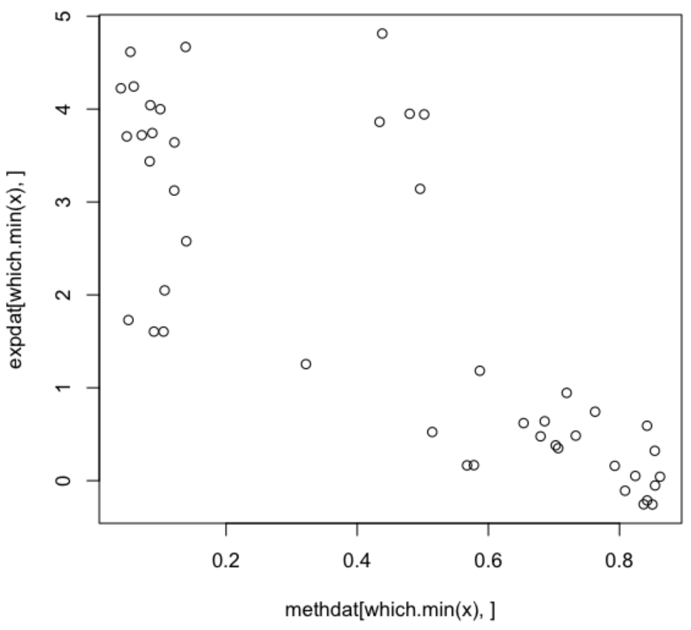
The same general trend seems to be present using the Spearman correlation coefficient as with the Pearson correlation coefficient.

```
> plot(density(y[!is.na(y)]), col = "green",
 +     main = "Spearman Correlation")
 > lines(density(x[!is.na(x)]), col = "red")
 > legend(0.3, 2, legend = c("expected",
 +     "observed"), col = c("green", "red"),
+       lty = 1)
```



And the plot looks hearly identical. Did we do any better with finding biologically meaningful top methylation candidates?

```
> plot(methdat[which.min(x), ], expdat[which.min(x),
+       ])
```



That looks better. The shape of the plot is interesting. Except for five data points, there appears to be a threshold effect whereby genes are more highly expressed only when methylation is very low.

- Try writing a short function that will take as input an integer that represents the rank in correlation (lowest being the first), the correlation vector, and the two data matrices and produce a plot of the data for that probe.
- Extend the function to include the gene name in the plot.
- Instead of returning the correlation coefficient like above, change the code so that a p.value is returned. *Hint*: try experimenting with the cor.test function.

Questions for thought:

- How would you go about determining if there is a regional bias in gene expression and DNA methylation?

3.2 CGH Data Analysis

CGH data are unlike many other types of high-throughput data such as gene expressin or DNA methylation. The fact that copy number in the genome is *piecewise continuous* is used by copy number segmentation methods to take measurements that are noisy at the individual probe level and *smooth* them in a non-continuous way; that is, these methods try to respect natural changepoints in the data. We will be using the DNAcopy package to facilitate some of these analyses. Make sure that it is installed.

```
> require(DNAcopy, quiet = TRUE)
```

If this doesn't work, go ahead and install using biocLite().

3.2.1 Quality control issues

I had already preprocessed the data above into a useful form. The data set is in the form of a limma MAList. The snapCGH package takes that as an input.

```
> library(limma)
 > require(DNAcopy)
 > data(cghTCGAMA)
 > class(cghTCGAMA)
 [1] "MAList"
attr(,"package")
 [1] "limma"
```

As you can see, the data just loaded are in the form of a limma MAList. The M values in this MAList are *not* gene expression measures, but log2 ratios of tumor DNA to a reference normal DNA. For normal individuals, the ratio of sample to reference signal will be close to 1 for a log2 ratio of 0. Gain of a copy of DNA at a locus will result in a value of log2(3/2) while loss of a copy will show log2(1/2). The sex chromosomes will behave slightly differently; that behavior will depend on the genders of both the reference and the sample.

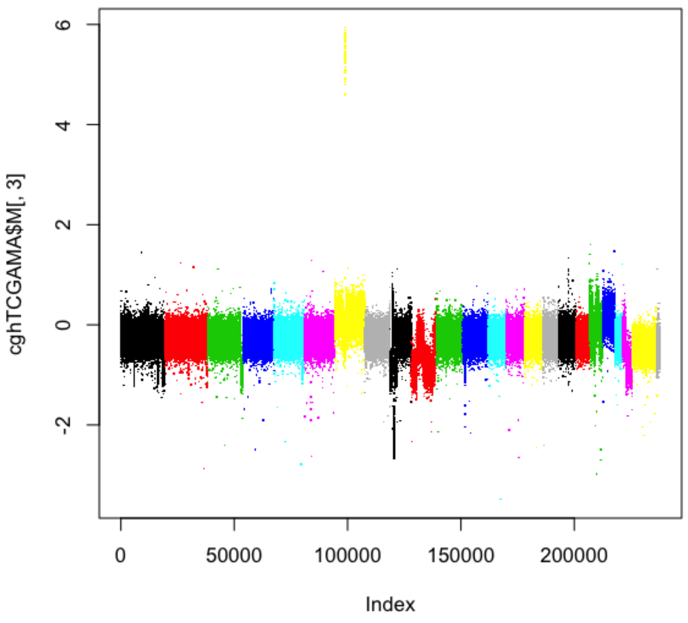
The annotation for these data are in the $genes list element.

```
> head(cghTCGAMA$genes)
       Row Col ProbeUID ControlType
 8437    19 233     8589           0
 25100   56  31    24444           0
 61506  135 419    61188           0
 48639  107 318    48398           0
 69696  153 403    69245           0
 107488 236 354   105927           0
            ProbeName                 GeneName
 8437     A_14_P112718                AK026901
 25100  A_16_P15000916                AK026901
 61506  A_16_P15001074                AK125248
 48639  A_16_P00000012 chr1:000736483-000736542
 69696  A_16_P00000014 chr1:000742533-000742586
 107488 A_16_P00000017 chr1:000746956-000747005
                 SystematicName
 8437   chr1:000554268-000554327
 25100  chr1:000554287-000554346
 61506  chr1:000639581-000639640
 48639  chr1:000736483-000736542
 69696  chr1:000742533-000742586
 107488 chr1:000746956-000747005
                                                Descrip-
tion
 8437      Homo sapi-
ens cDNA: FLJ23248 fis, clone COL03555.
```

```
 25100     Homo sapi-
ens cDNA: FLJ23248 fis, clone COL03555.
 61506  Homo sapi-
ens cDNA FLJ43258 fis, clone HHDPC1000001.
 48639                                                 Un-
known
 69696                                                 Un-
known
 107488                                                Un-
known
       chrom  start    end Chr Position
 8437    chr1 554268 554327   1   554268
 25100   chr1 554287 554346   1   554287
 61506   chr1 639581 639640   1   639581
 48639   chr1 736483 736542   1   736483
 69696   chr1 742533 742586   1   742533
 107488  chr1 746956 747005   1   746956
```

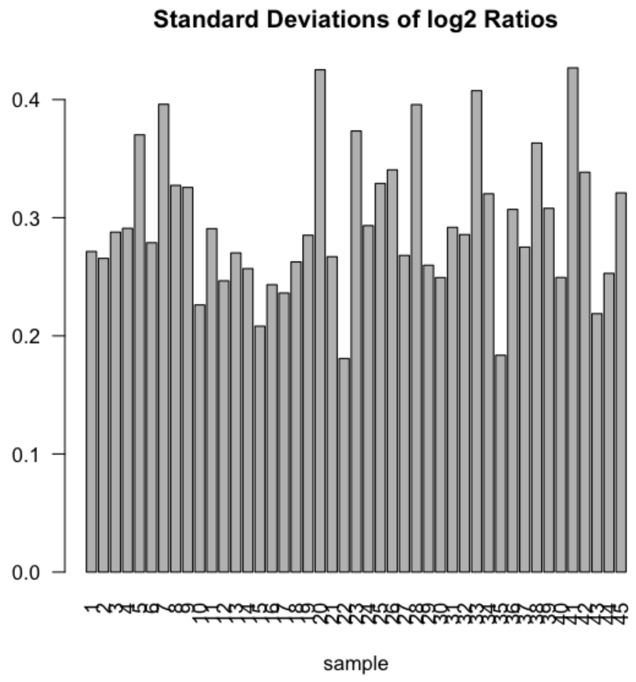
The data should be sorted by chromosome and position already, so we can plot a whole genome of log2 ratios easily with a single command. I add color to distinguish the chromosomes.

```
> plot(cghTCGAMA$M[, 3], col = cghTCGAMA$genes$Chr,
+      pch = ".")
```



You can see that most of the values are, indeed, near 0 with some spread. This spread represents the “noise” in the data and should probably be quantified. Normally, the spread of such values can be estimated by a measure such as the standard deviation. This will work just fine if the entire genome has a common mean (no copy number variation). How do the standard deviations look across our samples?

```
> log2sds = apply(cghTCGAMA$M, 2, sd)
 > par(las = 2)
 > barplot(log2sds, main = "Standard Devia-
tions of log2 Ratios",
 +     xlab = "sample", names.arg = 1:ncol(cghTCGAMA))
```

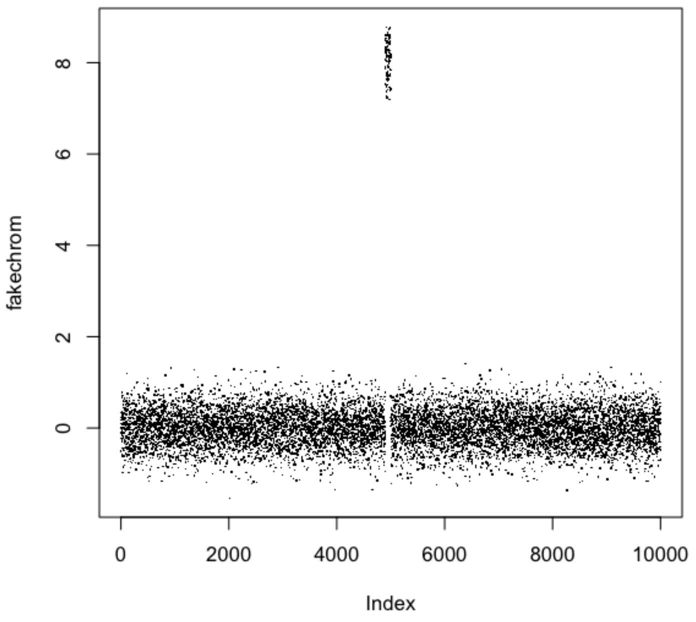


When advising experimentalists of the quality of their CGH arrays, it is important to be able to provide a robust estimate of the noise in the data. To see if the standard deviation is a very robust estimator of the noise. Let's consider a hypothetical chromosome with 10000 probes on it and no copy number variation. We can simulate such a thing quickly in R using rnorm and let's use a standard deviation of 0.4, similar to that in our data.

```
> fakechrom = rnorm(10000, sd = 0.4)
 > sd(fakechrom)
 [1] 0.3963893
```

Not very interesting, but let's put a copy number change in the middle of the chromosome that is similar in scale to that shown in the sample you plotted above:

```
> fakechrom[4900:5000] = fakechrom[4900:5000] +
 +     4
 > plot(fakechrom, pch = ".")
 > sd(fakechrom)
 [1] 0.5642358
```



Note that the standard deviation is now quite a bit larger than the 0.4 that we had anticipated even though we modified only 100 datapoints out of a total of 10,000. In other words, the standard deviation is sensitive to outliers in the data and with cancer samples, there can be quite a few outliers as many cancers have highly disordered and aneuploid genomes. With genomic data of any kind that might be expected to behave in a piecewise constant fashion, one can use another measure of noise, the derivative log ratio spread (DLRS). Defined here:

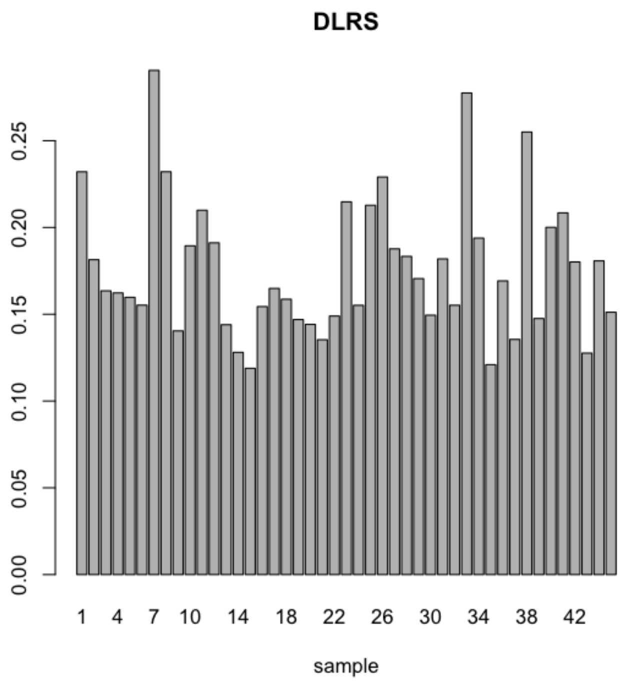
```
> dlrs <- function(x) {
 +     nx <- length(x)
 +     if (nx < 3) {
 +         stop("Vector length>2 needed for computa-
tion")
 +     }
 +     tmp <- embed(x, 2)
 +     diffs <- tmp[, 2] - tmp[, 1]
 +     dlrs <- IQR(diffs, na.rm = TRUE)/(sqrt(2) *
 +         1.34)
 +     return(dlrs)
 + }
```

What is the dlrs of our fake chromosome, even with the noise added?

```
> dlrs(fakechrom)
 [1] 0.3995698
```

And of our samples?

```
> log2dlrs = apply(cghTCGAMA$M, 2, dlrs)
 > barplot(log2dlrs, main = "DLRS", xlab = "sample",
 +     names.arg = 1:length(log2dlrs))
```



The DLRS is a useful measure of the actual noise in the data and is largely unaffected by “signal”. As such, it is a useful reporting tool for the technical quality of an array. All of the arrays in our sample set have DLRS values < 0.3, a maximum value recommended by Agilent.

We have applied the dlrs here to two-color Agilent CGH data, but such a measure could also be applied to log2 ratios from SNP arrays or even to copy number estimates generated by second-generation sequencing.

3.2.2 Data segmentation

A key component of making the best use of CGH data is called “segmentation”. There is extensive literature and literally dozens of methods for performing segmentation on CGH data. One that is particularly popular, easy to use, and fairly robust with not much need for parameter tuning on this data set is Circular Binary Segmentation (CBS) and is implemented in the DNAcopy package.

```
> require(DNAcopy)
 > citation("DNAcopy")
To cite package ‘DNAcopy’ in publications
use:

  Venkatraman E. Seshan and Adam Olshen ().
  DNAcopy: DNA copy number data analysis. R
  package version 1.27.1.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {DNAcopy: DNA copy number data analysis},
    author = {Venkatraman E. Seshan and Adam Olshen},
    note = {R package version 1.27.1},
  }

ATTENTION: This citation information has
been auto-generated from the package
DESCRIPTION file and may need manual
editing, see ‘help("citation")’ .
```

The citation function can be used with any R package to get a suggested reference or references. In some cases, the authors will list publications here, also. With the DNAcopy package loaded, we need to convert our MAList into a CNA object that the DNAcopy package understands. After doing so, we can proceed with the segmentation process.

```
> cna = CNA(cghTCGAMA$M, chrom = cghTCGAMA$genes$Chr,
 +     maploc = cghTCGAMA$genes$Position)
```

Warning messages are OK in the statements above. Error messages are not.

To keep the lab time to a minimum, I have already segmented the data and saved the results, so there is no need to run the next command. Feel free to do so if you like, but it will take about 5-10 minutes to complete.

```
> DNAcopyResult = segment(cna)
```

The interesting part of the DNAcopyResult object is the DNAcopyResult$output data.frame. Take a look at the first few rows to get a sense of what is in that data.frame:

```
> data(DNAcopyResult)
 > class(DNAcopyResult)
 [1] "DNAcopy"
 > str(DNAcopyResult)
List of 4
 $ data   :Classes ‘CNA’ and ’data.frame’:    237834 obs. of  47 vari-
ables:
  ..$ chrom    :Class ’AsIs’  int [1:237834] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ maploc   : int [1:237834] 554268 554287 639581 736483 742533 746956 757922 76959
  ..$ Sample.1 : num [1:237834] -0.835 -0.119 -0.303 -
0.848 -0.86 ...
  ..$ Sample.2 : num [1:237834] -0.397 -0.279 -0.256 -
0.435 -0.511 ...
  ..$ Sample.3 : num [1:237834] -0.307 -0.412 -0.389 -
0.197 -0.408 ...
  ..$ Sample.4 : num [1:237834] -0.186 -0.295 -
0.33 0.361 0.284 ...
```

```
  ..$ Sample.5 : num [1:237834] -0.462755 -1.208144 -
0.074967 -0.000619 -0.204713 ...
  ..$ Sample.6 : num [1:237834] -0.00601 -
0.40103 0.03126 0.09049 0.02756 ...
  ..$ Sample.7 : num [1:237834] -0.677 -0.257 -0.303 -
0.44 -0.788 ...
  ..$ Sample.8 : num [1:237834] -0.4865 -0.309 -0.0106 -
0.1609 -0.7247 ...
  ..$ Sample.9 : num [1:237834] -0.46 -0.2652 0.0353 -
0.0376 -0.0319 ...
  ..$ Sample.10: num [1:237834] -0.3808 -0.1247 0.0954 -
0.3125 -0.6278 ...
  ..$ Sample.11: num [1:237834] -0.7593 -0.4938 -
0.2141 -0.0964 -0.4308 ...
  ..$ Sample.12: num [1:237834] -0.122 -0.295 -0.166 -
0.595 -0.438 ...
  ..$ Sample.13: num [1:237834] -0.337 -0.251 -0.202 -
0.28 -0.413 ...
  ..$ Sample.14: num [1:237834] -1.0474 -
0.9955 0.5392 0.1417 0.0108 ...
  ..$ Sample.15: num [1:237834] -
0.0695 0.2873 0.4508 0.1924 0.2408 ...
  ..$ Sample.16: num [1:237834] -0.2021 -0.3931 -
0.3783 -0.0726 -0.4625 ...
  ..$ Sample.17: num [1:237834] -0.04025 -
0.19164 0.35159 0.00462 -0.37364 ...
  ..$ Sample.18: num [1:237834] -0.0651 -0.0718 0.0966 -
0.0595 -0.3145 ...
  ..$ Sample.19: num [1:237834] -0.4576 -
1.4009 0.1637 0.1842 -0.0925 ...
  ..$ Sample.20: num [1:237834] -0.942 -1.675 0.139 -
0.287 -0.15 ...
  ..$ Sample.21: num [1:237834] -0.0886 -
0.038 0.2081 0.1817 0.4394 ...
  ..$ Sample.22: num [1:237834] -
0.0676 0.1388 0.3111 0.0152 0.0574 ...
  ..$ Sample.23: num [1:237834] -0.5645 -0.0665 0.0776 -
0.4033 -0.462 ...
  ..$ Sample.24: num [1:237834] -0.8593 -1.9658 0.0818 -
0.0369 0.0117 ...
  ..$ Sample.25: num [1:237834] -0.2496 -0.2859 -
0.022 0.0901 -0.2358 ...
  ..$ Sample.26: num [1:237834] -
0.16957 0.42437 0.48651 -0.00381 -0.35216 ...
  ..$ Sample.27: num [1:237834] -
0.0519 0.1894 0.3673 0.1334 -0.0929 ...
  ..$ Sample.28: num [1:237834] -0.21457 -
0.2359 0.23507 0.00635 -0.20374 ...
  ..$ Sample.29: num [1:237834] 0.0599 -
0.029 0.3558 0.1296 0.0698 ...
```

```
  ..$ Sample.30: num [1:237834] -0.2131 -0.1149 0.0755 -
0.4354 -0.3781 ...
  ..$ Sample.31: num [1:237834] 0.0649 -
0.4275 0.5949 0.1539 -0.1956 ...
  ..$ Sample.32: num [1:237834] -0.149 -
0.155 0.311 0.175 -0.317 ...
  ..$ Sample.33: num [1:237834] -0.363 -0.62 -
0.155 0.102 -0.571 ...
  ..$ Sample.34: num [1:237834] -0.39793 -0.75989 -
0.00638 -0.32363 -0.75243 ...
  ..$ Sample.35: num [1:237834] -0.678 -0.643 0.229 -
0.115 -0.169 ...
  ..$ Sample.36: num [1:237834] -0.479 -0.858 0.221 -
0.585 -0.602 ...
  ..$ Sample.37: num [1:237834] -0.4255 -
0.4983 0.4769 0.0751 -0.1221 ...
  ..$ Sample.38: num [1:237834] -1.147 -1.175 -0.903 -
0.274 -1.228 ...
  ..$ Sample.39: num [1:237834] 0.17 0.1706 0.169 -
0.0297 -0.0114 ...
  ..$ Sample.40: num [1:237834] -0.681 -0.845 -1.005 -
0.655 -0.893 ...
  ..$ Sample.41: num [1:237834] -0.5468 0.0712 -0.2086 -
0.416 -0.5412 ...
  ..$ Sample.42: num [1:237834] -0.56146 0.00593 -
0.0324 -0.30969 -0.57682 ...
  ..$ Sample.43: num [1:237834] -0.7518 -1.5594 0.5325 -
0.0239 -0.0884 ...
  ..$ Sample.44: num [1:237834] -0.47 -
0.3217 0.3012 0.1977 -0.0167 ...
  ..$ Sample.45: num [1:237834] -0.316 -
0.508 0.251 0.143 -0.358 ...
  ..- attr(*, "data.type")= chr "logratio"
 $ output :'data.frame':      7052 obs. of  6 variables:
  ..$ ID       : chr [1:7052] "Sample.1" "Sam-
ple.1" "Sample.1" "Sample.1" ...
  ..$ chrom    : int [1:7052] 1 1 1 1 1 2 2 2 3 4 ...
  ..$ loc.start: int [1:7052] 554268 202195769 203399457 246794522 246880850 20341 345
  ..$ loc.end  : int [1:7052] 202172219 203393803 246755225 246874992 247190718 345397
  ..$ num.mark : num [1:7052] 15576 127 3666 12 33 ...
  ..$ seg.mean : num [1:7052] -0.584 1.653 -0.605 -
1.217 -0.531 ...
 $ segRows:'data.frame':      7052 obs. of  2 variables:
  ..$ startRow: int [1:7052] 1 15577 15704 19370 19382 19415 22138 22140 38086 54092
  ..$ endRow  : int [1:7052] 15576 15703 19369 19381 19414 22137 22139 38085 54091 662
 $ call   : language segment(x = cna, verbose = 2)
 - attr(*, "class")= chr "DNAcopy"
 > head(DNAcopyResult$output)
        ID chrom loc.start   loc.end num.mark
 1 Sample.1     1    554268 202172219    15576
```

```
2 Sample.1     1 202195769 203393803      127
3 Sample.1     1 203399457 246755225     3666
4 Sample.1     1 246794522 246874992       12
5 Sample.1     1 246880850 247190718       33
6 Sample.1     2     20341  34539740     2723
 seg.mean
1  -0.5840
2   1.6526
3  -0.6045
4  -1.2169
5  -0.5308
6  -0.5947
```

Each row in this data.frame describes a segment of the genome that, statistically speaking, has a constant copy number. The ID and chrom columns are self-explanatory. The loc.start, loc.end, num.mark, and seg.mean simply describe the chromosome location for the start and end of the copy-number-stable region, the number of probes on the array that are represented in the region, and the mean log2 ratio of that region. The data in this form are great for loading into a database or for some visualization tools that require segmented data. For the purposes of this lab, we might want things back in a matrix-like form where the rows represent segmental mean for the each of the probes and the columns represent samples. The following little code chunk will get us there:

```
> dnacPerSample = split(DNAcopyResult$output,
 +     DNAcopyResult$output$ID)
 > dnacPerSample = dnacPerSam-
ple[order(as.numeric(sub("Sample.",
 +     "", names(dnacPerSample))))]
 > dnacDataMat = do.call(cbind, lapply(dnacPerSample,
 +     function(z) {
 +         return(rep(z[, 6], z[, 5]))
 +     }))
 > dim(dnacDataMat)
 [1] 237834     45
```

To keep the probe annotation information and the data together, we can use the ExpressionSet class.

```
> require(Biobase)
 > pdata = new("AnnotatedDataFrame", cghTCGAMA$targets)
 > sampleNames(pdata) = cghTCGAMA$targets$sample
 > colnames(dnacDataMat) = cghTCGAMA$targets$sample
 > dnacEset = new("ExpressionSet", phenoData = pdata,
 +     exprs = dnacDataMat, feature-
Data = new("AnnotatedDataFrame",
 +         cghTCGAMA$genes))
```

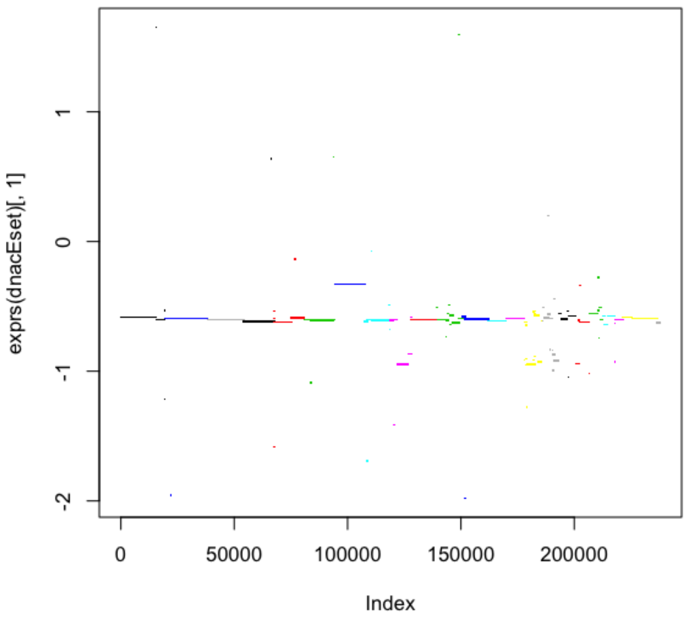
At this point, you have successfully segmented (smoothed) your data and created a not-so-aptly-named ExpressionSet.

3.2.3 Centering the CGH data

Remember that our goal is to find regions of gain and loss. To do so, we want to make sure that the "baseline" for all the samples is near a log2 ratio of zero. I include "baseline" in quotes because the baseline represents our estimate of the "normal" copy number state. An assumption that I and others will make is that the best estimate of the "normal" state is that which minimizes the number of probes that are not near the log2 ratio of 0. Perhaps a visual representation is useful here. For this part of the exercise, we will be using DNAcopy segmented values that we prepared earlier.

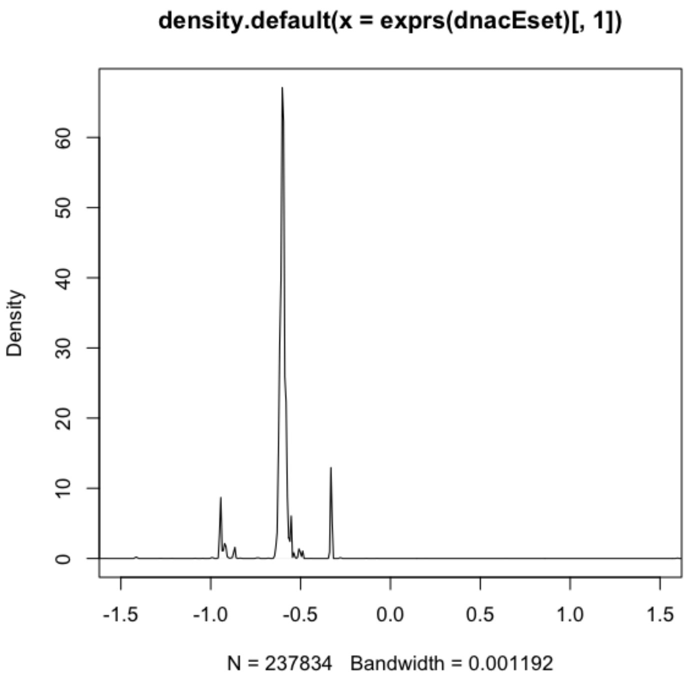
And plot one example where the center is probably not correct:

```
> plot(exprs(dnacEset)[, 1], pch = ".",
+     col = fData(dnacEset)$chrom)
```



Notice that the "center" of the distribution is at log2 ratio of -0.6. How can we reliably and robustly find the "center" of this distribution? Let's look at a density plot of the *segmented* data.

```
> plot(density(exprs(dnacEset)[, 1]), xlim = c(-1.5,
+     1.5))
```



The mode of this distribution is exactly what we are interested in using as the "center". Therefore, I include a little function to find the mode of a vector of numbers.

```
> findMode = function(z) {
+     tmpdens = density(z)
+     return(tmpdens$x[which.max(tmpdens$y)])
+ }
```

We can apply it to all samples simultaneously using the apply functionality of R.

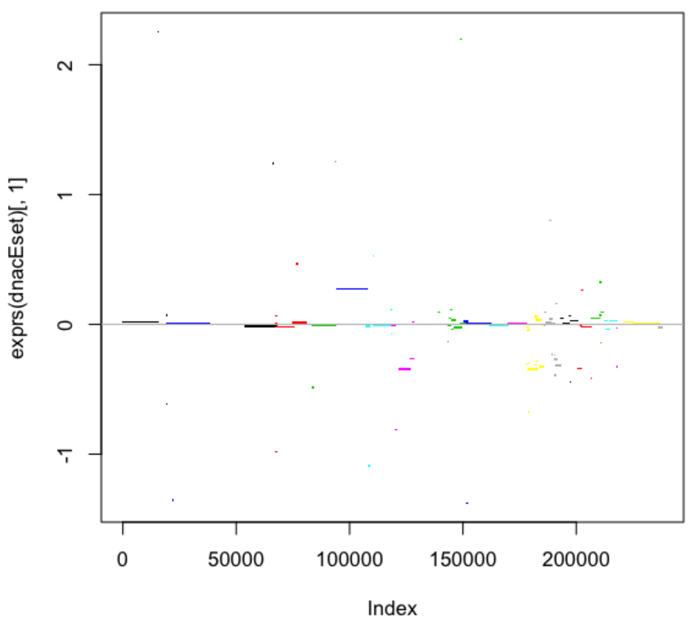
```
> ctrs = apply(exprs(dnacEset), 2, findMode)
> summary(ctrs)
    Min.  1st Qu.   Median     Mean  3rd Qu. 
-0.70930 -0.28200 -0.12780 -0.16060 -0.02138 
    Max. 
 0.16300 
```

To "fix" the data (that is, subtract the offsets from the data to appropriately "center" them), we can use the sweep function to add the appropriate offset to the data to get the "baseline" near the log2 of 0, our goal.

```
> exprs(dnacEset) = sweep(exprs(dnacEset), 
+     2, ctrs)
```

Now, if we plot the same sample as before, it should show a baseline close to log2 of 0.

```
> plot(exprs(dnacEset)[, 1], pch = ".",
+     col = fData(dnacEset)$chrom)
> abline(h = 0, col = "grey")
```



Now, on to biology, finally!

3.2.4 Global Copy Number Behavior

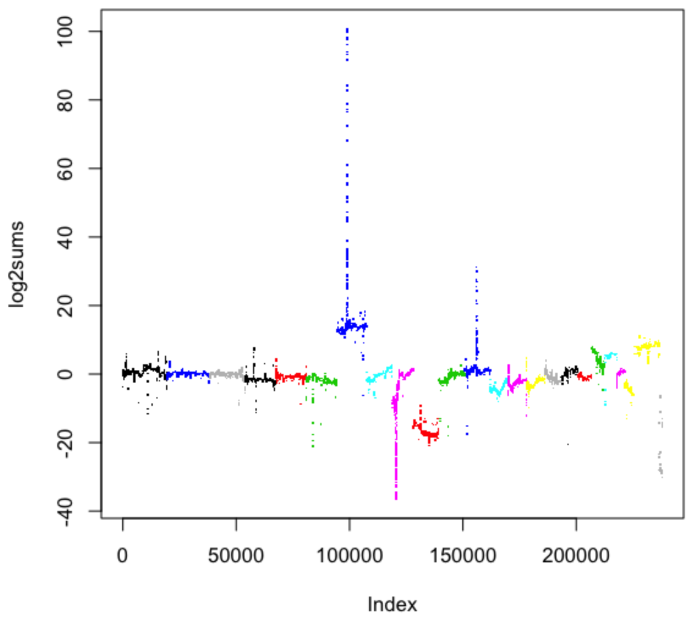
Now that we have performed segmentation, thereby smoothing the data while maintaining the natural breakpoints in the data, we want to look for the regions with the highest proportion of copy number changes in the samples. Particularly if we successfully remove common copy number variants that are present in normal individuals, such regions may be thought to be the most interesting biologically.

There are a number of methods described in the literature to find such ”interesting” regions. However, we will keep it simple here and try just a couple. Using our segmented and centered data, let’s simply add up the log2 ratios at each locus for every sample.

```
> log2sums = rowSums(exprs(dnacEset))
> length(log2sums)
[1] 237834
```

And plot the result.

```
> plot(log2sums, pch = ".", col = fData(dnacEset)$chrom)
```



This figure is very important to understand. The distance away from the y=0 line, here simply the sum of the segmented log2 ratios, is somewhat a measure of biological importance. Recall that regions that are gained may contain oncogenes while regions with loss are potentially harboring tumor suppressors. A few regions of interest immediately pop out. There are small, high excursions on chromosomes 7 and 12. There is a significant low excursion on chromosome 9 (and also chromosome Y, which we will choose to ignore, but why would chromosome Y appear to be deleted?). Also, though lower excursion, the baseline for chromosome 7 shows a gain while the baseline for chromosome 10 shows a loss. Does this make sense biologically given what is known about glioblastoma?

This is all well-and-good, but what genes are in the regions of high gain and loss? Well, let's just ask the data. Let's find the probes that correspond to the regions of highest gain and lowest loss. To do so, I will use a simple threshold and pull out the probes that have log2sums higher than the threshold. In these data, the thresholds are quite easy to choose; this is not, in general, the case.

```
> highcopyprobeIdx = log2sums > 20
 > highcopyprobes = fData(dnacEset)[highcopyprobeIdx,
 +     c("chrom", "GeneName")]
 > highcopygenes = unique(highcopyprobes[-grep("chr",
 +     highcopyprobes$GeneName), ])
 > highcopygenes[, c("chrom", "GeneName")]
       chrom GeneName
 210431   chr7 FLJ45974
 65369    chr7    VSTM2
 174544   chr7 CR613464
 140801   chr7 BC045679
```

```
241146  chr7   SEC61G
165548  chr7     EGFR
44942   chr7   K03193
91766   chr7   LANCL2
93272   chr7 AK128355
121710  chr7     ECOP
125336  chr7 BC015339
192911  chr7 FLJ44060
209927  chr7 BC094796
130609  chr7   ZNF713
135653  chr7 CR590495
155510  chr7   MRPS17
156592  chr7     GBAS
29164   chr7     PSPH
210171  chr7    CCT6A
225314  chr7    SUMF2
119849  chr7    PHKG1
1910    chr7   Y10275
190999  chr7   CHCHD2
128900  chr7 AL713776
37842   chr7 BC035176
193160 chr12      OS9
43012  chr12   CENTG1
205540 chr12  TSPAN31
72692  chr12     CDK4
3017   chr12   MARCH9
67423  chr12 AK093897
85074  chr12  CYP27B1
195380 chr12   METTL1
155592 chr12  FAM119B
158880 chr12     TSFM
135524 chr12     AVIL
```

For those not in the cancer field, I point out two important cancer genes. The high copy number region on chromosome 7 is harboring an EGFR amplification. Just check the literature for EGFR amplification in glioblastoma! On chromosome 12, CDK4 is in the amplified region and is an important cell cycle regulator.

What about the region of loss?

```
> lowcopyprobeIdx = log2sums < (-30)
> lowcopyprobes = fData(dnacEset)[lowcopyprobeIdx,
+     c("chrom", "GeneName")]
> lowcopygenes = unique(lowcopyprobes[-grep("chr",
+     lowcopyprobes$GeneName), ])
> lowcopygenes[, c("chrom", "GeneName")]
       chrom GeneName
81796   chr9     MTAP
107171  chr9 AF109294
166419  chr9   CDKN2A
41673   chr9   CDKN2B
```

```
10694    chrY    RBMY1F
```

Again, for non-cancer folks, CDKN2A is a well-known tumor suppressor!

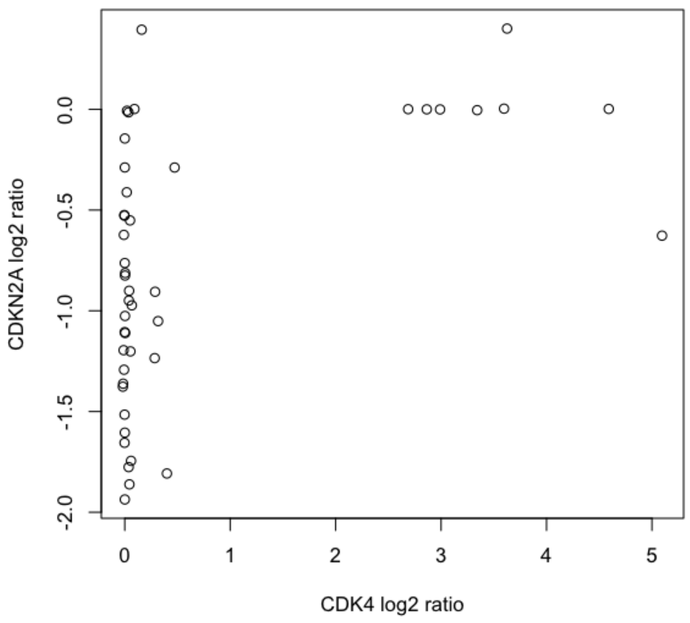
3.3 Follow up on interesting finding

So, we have three important cancer genes popping up by basic visual inspection! But, one might begin to wonder a bit about the relationship between these genes. In fact, since there were only three genes that were viable candidates and EGFR was a well-known finding in glioblastoma, I decided to dig a bit further with CDK4 and CDKN2A.

I'll leave it as an exercise to the reader to learn extensively about these two genes and their interactions, but suffice it to say that the full gene name of CDKN2A is "cyclin-dependent kinase inhibitor 2A (melanoma, p16, inhibits CDK4)". CDK4 drives the cell cycle, roughly, and CDKN2A serves to negatively regulate that effect. CDK4 is the prototypic oncogene and CDKN2A is the prototypic tumor suppressor, suppressing tumors by negatively regulating CDK4.

Back to the data, we want to look at the relationship between copy number estimates of CDKN2A and CDK4. I will choose representative probes for each gene and look at a plot of them together.

```
> cdk4probeidx = which(fData(dnacEset)$GeneName ==
+     "CDK4")[1]
> cdkn2aprobeidx = which(fData(dnacEset)$GeneName ==
+     "CDKN2A")[1]
> plot(exprs(dnacEset)[cdk4probeidx, ],
+     exprs(dnacEset)[cdkn2aprobeidx, ],
+     xlab = "CDK4 log2 ratio", ylab = "CDKN2A log2 ratio")
```



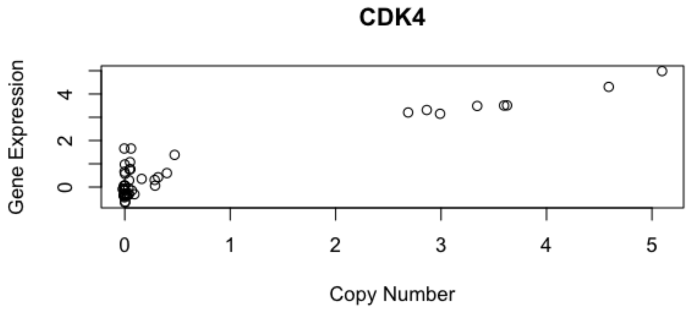
How would you interpret the plot? Does this make sense given what you know about CDKN2A and CDK4 biology?

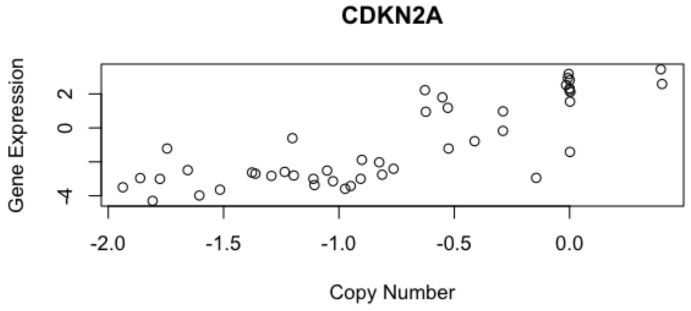
Perhaps we want to look at the effect of copy number on gene expression for these genes. To do so, we need the copy number data to be in the same order as the gene expression data.

```
> cnCDK4 = exprs(dnacEset)[cdk4probeidx,
 +     order(sampleNames(dnacEset))]
 > cnCDKN2A = exprs(dnacEset)[cdkn2aprobeidx,
 +     order(sampleNames(dnacEset))]
 > exCDK4 = exprs(expTCGA)["CDK4", or-
der(sampleNames(expTCGA))]
 > exCDKN2A = exprs(expTCGA)["CDKN2A", order(sampleNames(expTCGA))]
```

And now, we can make a plot of the two genes to see how things look:

```
> par(mfrow = c(2, 1))
 > plot(cnCDK4, exCDK4, main = "CDK4", xlab = "Copy Num-
ber",
 +     ylab = "Gene Expression")
 > plot(cnCDKN2A, exCDKN2A, main = "CDKN2A",
 +     xlab = "Copy Number", ylab = "Gene Expression")
```





And how about methylation, particularly of CDKN2A, since that could be a secondary mechanism for silencing the gene, besides gene deletion? We'll need to get at least one methylation data vector or CDKN2A.

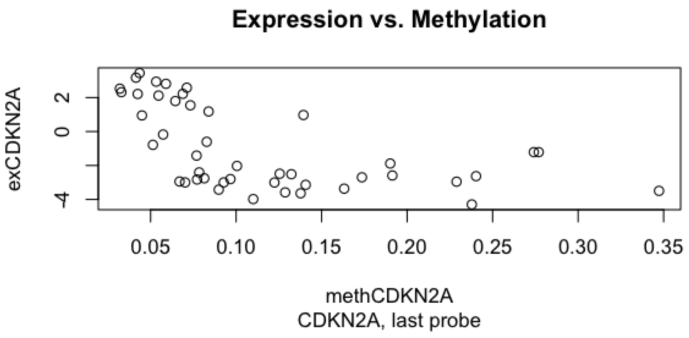
```
> methGeneNames = unlist(mget(featureNames(methTCGA),
 +     IlluminaHumanMethylation27kSYMBOL,
 +     ifnotfound = NA))
 > cdkn2amethidx = which(methGeneNames ==
 +     "CDKN2A")
 > cor(t(betas(methTCGA)[cdkn2amethidx, ]))
           cg00718440 cg03079681 cg13479669
cg00718440  1.0000000 -0.1392268 -0.1244518
cg03079681 -0.1392268  1.0000000  0.7301408
cg13479669 -0.1244518  0.7301408  1.0000000
cg26673943 -0.1494681  0.8510897  0.7594127
           cg26673943
cg00718440 -0.1494681
cg03079681  0.8510897
cg13479669  0.7594127
cg26673943  1.0000000
```

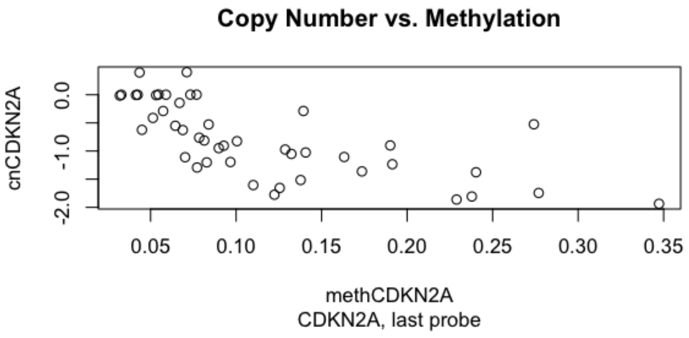
Note that three out of four probes are correlated positively with each other. So, we can exclude the first probe because it does not agree with the other three. This might be an incorrect assumption, but we can always check later. Using the last probe, arbitrarily, as representative, we can now interrogate the effect of methylation on gene expression of CDKN2A.

```
> methCDKN2A = betas(methTCGA)["cg26673943",
 +     order(sampleNames(methTCGA))]
```

Finally, let's plot against gene expression and copy number:

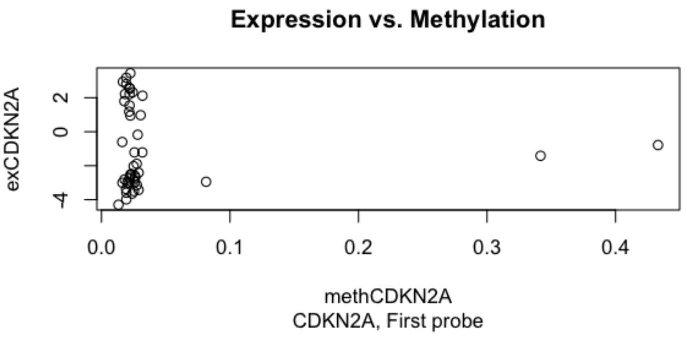
```
> par(mfrow = c(2, 1))
 > plot(methCDKN2A, exCDKN2A, main = "Expres-
sion vs. Methylation",
 +     sub = "CDKN2A, last probe")
 > plot(methCDKN2A, cnCDKN2A, main = "Copy Num-
ber vs. Methylation",
 +     sub = "CDKN2A, last probe")
```

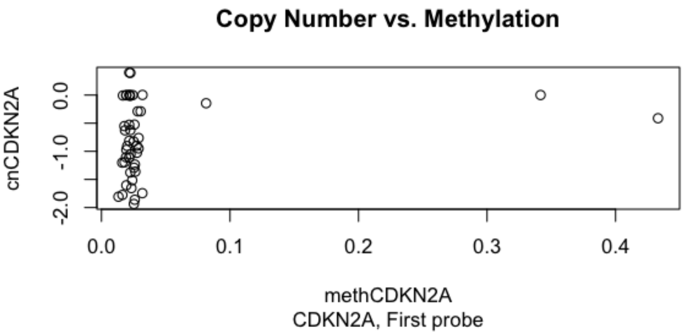




Taking the expression plot, first, we see a very nice negative correlation between expression and methylation, despite the fact that the maximal observed methylation is 0.3 or so. Looking at the bottom plot, however, it appears that the same correlation holds with copy number. We might hope to see that the normal copy number samples would be more likely to have higher methylation values. Instead, the opposite is true. Remember that we chose to ignore the first methylation probe because it was poorly correlated with the other three methylation probes. Let's revisit that probe.

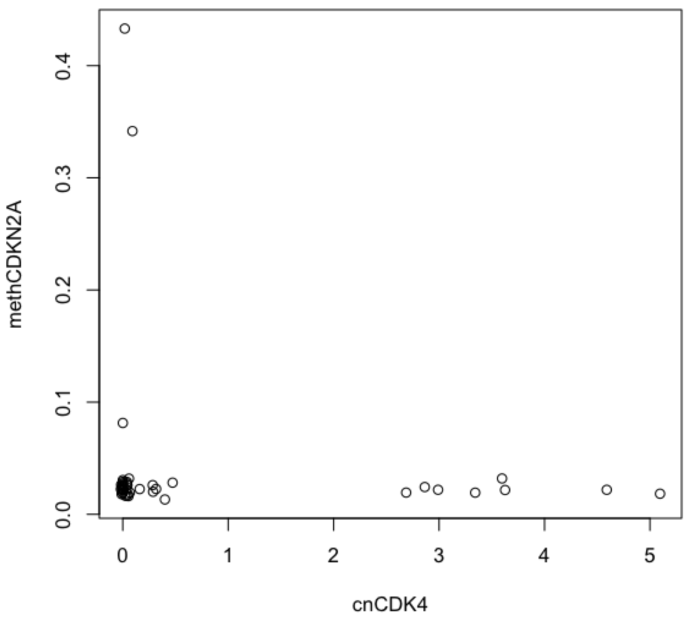
```
> methCDKN2A = betas(methTCGA)["cg00718440",
 +     order(sampleNames(methTCGA))]
 > par(mfrow = c(2, 1))
 > plot(methCDKN2A, exCDKN2A, main = "Expres-
sion vs. Methylation",
 +     sub = "CDKN2A, First probe")
 > plot(methCDKN2A, cnCDKN2A, main = "Copy Num-
ber vs. Methylation",
+     sub = "CDKN2A, First probe")
```





While the number of samples with high methylation is small (probably just 2), these samples show lowish CDKN2A gene expression levels. With regard to copy number, the samples with high methylation are, indeed, those without deletion, suggesting that methylation could be responsible for CDKN2A silencing in these samples. And what about the CDK4 amplification status of these samples?

```
> plot(cnCDK4, methCDKN2A)
```



These two samples have stone-cold normal CDK4 copy number.

4 Conclusions

We have performed a subset of a data integration exercise on TCGA glioblastoma data and have an incredibly interesting finding within the space of an afternoon. There is a small but measurable global effect of methylation on gene expression. Also, we find in the copy number data the most common amplification, EGFR, as well as gain of chromosome 7 and loss of chromosome 10, also common findings in glioblastoma.

At a more mechanistic level, it seems that the relationship between CDKN2A, acting as tumor suppressor and it's target, CDK4 is quite clear in these data. We have a working hypothesis that CDK4 overexpression due to amplification or CDKN2A underexpression due to either DNA loss (deletion) or methylation at one CpG site are approximately mutually exclusive and likely represent the same pathway being disregulated in these data.

The TCGA project provides a nice testbed for these types of data integration exercies. MicroRNA and sequence variation data are also available for these samples, making the potential for finding even more interesting biology--even with just a little work--an attractive proposition. Note that I have not tried to be very sophisticated in these analyses and I do not claim that the analysis here is anything more than a data exploration exercise. Obviously, more sophisticated techniques including dimensionality reduction, network creation, and formal data integration techniques would likely enhance the value of the dataset significantly.

5 sessionInfo

```
> sessionInfo()
R Under development (unstable) (2011-08-01 r56587)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

locale:
 [1] en_US.utf-8/en_US.utf-8/en_US.utf-8/C/en_US.utf-
8/en_US.utf-8

attached base packages:
 [1] stats     graphics  grDevices utils
 [5] datasets  methods   base

other attached packages:
 [1] TCGAGBM_1.1
 [2] DNAcopy_1.27.1
 [3] IlluminaHumanMethylation27k.db_1.4.6
 [4] org.Hs.eg.db_2.5.0
 [5] RSQLite_0.9-4
 [6] DBI_0.2-5
 [7] AnnotationDbi_1.15.9
 [8] limma_3.9.11
 [9] methylumi_1.9.0
 [10] Biobase_2.13.7
 [11] ascii_1.4
 [12] proto_0.3-9.2

loaded via a namespace (and not attached):
 [1] annotate_1.31.0 grid_2.14.0
 [3] lattice_0.19-31 tools_2.14.0
 [5] xtable_1.5-6
```